1   **TITLE**

2   **METHOD FOR COMMONLY CONTROLLING DEVICE DRIVERS**

3   **CLAIM OF PRIORITY**

4   **[0001]**   This application makes reference to, incorporates the same herein, and claims all benefits

5   accruing under 35 U.S.C. §119 from an application for *COMMON CONTROL IMPLEMENT*

6   *METHOD FOR DEVICE DRIVER* earlier filed in the Korean Industrial Property Office on 8 August

7   2002 and there duly assigned Serial No. 2002-46757.

8   **BACKGROUND OF THE INVENTION**

9   **Field of the Invention**

10   **[0002]**   The present invention relates to a communication system, and more particularly to a

11   method for commonly controlling device drivers of a hardware chip level.

12   **Description of the Related Art**

13   **[0003]**   Currently, manufacturers of higher-order applications and lower-order device drivers

14   develop their products independently of one another. Presently, different vendors provide a large

15   number of device drivers. This means that the lack of communication between the manufacturers

16   of the higher-order applications and the lower-order device drivers can become serious.

17   Conventionally, APIs (Application Program Interfaces) are made on the basis of individual unique

18   respective standards of the higher-order applications and the lower-order device drivers.

1   Accordingly, the higher-order applications and the lower-order device drivers can be made to

2   conform to the standards of each other.

3   **[0004]**   Where the device driver cannot meet the requirements of the higher-order applications,

4   there is a problem in that a code or structure affecting the higher-order applications should be

5   corrected. At this time, the higher-order applications and the device driver should be re-verified.

6   **[0005]**   If a specific portion is changed where a first device is changed to a second device in

7   relation to a common application or two applications use a common device driver, the higher-order

8   applications and the device drivers should be re-verified. The re-verification requirement can

9   adversely affect the development of products and reduce competitiveness of the product because of

10   the increased time needed for the product development.

11   **SUMMARY OF THE INVENTION**

12   **[0006]**   Therefore, the present invention has been made in view of the above and other problems,

13   and it is an object of the present invention to provide a method capable of independently and

14   commonly employing a higher-order application hierarchy and a lower-order device driver by

15   arranging a DIA (Device Independent Access) hierarchy as an intermediate hierarchy between the

16   higher-order application hierarchy and the lower-order device driver for the sake of establishing a

17   mutual interface in a communication system.

1   **[0007]** It is another object of the present invention to provide a method capable of independently

2   and commonly employing a lower-order device driver hierarchy in a communication system.

3   **[0008]** It is yet another object of the present invention to provide a method capable of

4   independently and commonly employing a higher-order application hierarchy in a communication

5   system.

6   **[0009]** In accordance with an aspect of the present invention, the above and other objects can be

7   accomplished by the provision of a method for commonly controlling device drivers, including:

8   arranging a DIA (Device Independent Access) hierarchy between an application hierarchy and a

9   device driver hierarchy and applying a standardized rule of the DIA hierarchy to the application

10   hierarchy and the device driver hierarchy; and allowing the application hierarchy and the device

11   driver hierarchy to access the device driver hierarchy and the application hierarchy through the

12   standardized rule of the DIA hierarchy, respectively.

13   **BRIEF DESCRIPTION OF THE DRAWINGS**

14   **[0010]** A more complete appreciation of the invention, and many of the attendant advantages

15   thereof, will be readily apparent as the same becomes better understood by reference to the following

16   detailed description when considered in conjunction with the accompanying drawings in which like

17   reference symbols indicate the same or similar components, wherein:

18   **[0011]** Fig. 1 is a view illustrating a case where a device A is changed to a device B in relation to

1 a common application;

2 **[0012]** Fig. 2 is a view illustrating a case where two applications use a common device driver;

3 **[0013]** Fig. 3 is a view illustrating a concept of commonly controlling device drivers in accordance

4 with an embodiment of the present invention;

5 **[0014]** Fig. 4 is a view illustrating an access for device driver control commonization;

6 **[0015]** Fig. 5 is a view illustrating a connection structure of a DCB (Device Control Block) after

7 an API (Application Program Interface) "Dia_InitDevice" is called for device initialization;

8 **[0016]** Fig. 6 is a view illustrating a final DCB in a level 1 initialization stage;

9 **[0017]** Fig. 7 is a view illustrating a connection structure of a DCB when an HDLC (High-level

10 Data Link Control) device having four ports corresponding to a level 2 is initialized;

11 **[0018]** Fig. 8 is a view illustrating a connection structure of a DCB when a channel is initialized;

12 **[0019]** Fig. 9 is a view illustrating a structure of an event table;

13 **[0020]** Fig. 10 is a view illustrating a case where a device driver is changed (or replaced) in

14 accordance with an embodiment of the present invention;

15 **[0021]** Fig. 11 is a view illustrating a case where a higher-order application (program) is changed

16 in a conventional art;

17 **[0022]** Fig. 12 is a view illustrating a case where a lower-order device is changed in a

18 conventional art;

19 **[0023]** Fig. 13 is a view illustrating a case where a higher-order application (program) is changed

20 in accordance with an embodiment of the present invention;

21 **[0024]** Fig. 14 is a view illustrating a case where a lower-order device is changed in accordance

1   with an embodiment of the present invention; and

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

3   **[0025]**   Turning now to the drawings, Fig. 1 illustrates a case where a device A is changed to a

4   device B in relation to a common application. Fig. 2 illustrates a case where two applications use

5   a common device driver.

6   **[0026]**   Referring to Fig. 1, when the device A is changed to the device B in relation to a common

7   application 2 as a higher-order application 2, a device-A driver 4 is also changed to a device-B driver

8   6, wherein the device-A and device-B drivers 4 and 6 are lower-order device drivers. If so, APIs

9   have different characteristics associated with the devices A and B. Accordingly, the higher-order

10 application 2 should be also changed according to the changed device and device driver. When the

11 device driver is changed, all APIs associated with the changed lower-order device driver should be

12 newly verified. In other words, it should be verified whether the APIs appropriately operate on the

13 basis of the changed device driver. Further, it should be checked whether all higher-order

14 applications associated with the changed lower-order device driver affect others.

15 **[0027]**   Next, referring to Fig. 2, a higher-order application-A 10 and a higher-order application-B

16 12 use a common device driver 14. The higher-order applications-A and B 10 and 12 need different

17 formats from the device driver 14. For example, the higher-order application-A 10 needs two

18 formats A-1 and A-2 provided from the device driver 14. Otherwise, the higher-order application-B

1   12 needs three formats B-1, B-2 and B-3 provided from the device driver 14. This case occurs

2   because a format agreement between the applications has not been agreed upon. Accordingly, the

3   device driver 14 should change formats in response to different requirements from the higher-order

4   applications-A and B 10 and 12 and therefore the APIs should be changed and added. Where the

5   device driver cannot meet the requirements of the higher-order applications, there is a problem in

6   that a code or structure affecting the higher-order applications should be corrected. At this time, the

7   higher-order applications and the device driver should be re-verified.

8   [0028]   As described in connection with Figs. 1 and 2, if a specific portion is changed where the

9   device A is changed to the device B in relation to the common application or the two applications

10 use the common device driver, the higher-order applications and the device drivers should be

11 re-verified. The re-verification requirement can adversely affect the development of products and

12 reduce competitiveness of the product because of the increased time needed for the product

13 development.

14 [0029]   The present invention prevents a higher-order application hierarchy and a lower-order

15 device driver hierarchy from directly accessing each other by arranging a DIA (Device Independent

16 Access) hierarchy between the higher-order application hierarchy and the lower-order device driver

17 hierarchy in a communication system. Accordingly, the higher-order application hierarchy and the

18 lower-order device driver hierarchy can access the lower-order device driver hierarchy and the

19 higher-order application hierarchy via the DIA on the basis of the DIA's standardized rule,

1   respectively. Because the higher-order application hierarchy and the lower-order device driver

2   hierarchy accesses the lower-order device driver hierarchy and the higher-order application hierarchy

3   on the basis of the DIA's standardized rule, respectively, a period of time required for the

4   development of products, and costs of product development can be reduced, and efficiency of

5   product development can be improved.

6   [0030]   Now, preferred embodiments of the present invention will be described in detail with

7   reference to the annexed drawings. In the following description, a detailed description of known

8   functions and configurations incorporated herein will be omitted when it may make the subject

9   matter of the present invention rather unclear.

10   [0031]   Fig. 3 is a view illustrating a concept of commonly controlling device drivers in accordance

11   with an embodiment of the present invention. Referring to Fig. 3, a DIA (Device Independent

12   Access) hierarchy 22 is arranged between a higher-order application 20 and device drivers 24 and ·

13   26 in a communication system in accordance with the embodiment of the present invention. The

14   higher-order application 20 accesses the device drivers 24 and 26 via the DIA hierarchy 22 on the

15   basis of a standardized rule of the DIA hierarchy 22. Similarly, the device drivers 24 and 26 access

16   the higher-order application 20 via the DIA hierarchy 22 on the basis of the standardized rule of the

17   DIA hierarchy 22. This will be described in detail with reference to Fig. 4.

18   [0032]   Fig. 4 is a view illustrating an access for device driver control commonization. Referring

1 to Fig. 4, when the higher-order application 20 requests the DIA hierarchy 22 to provide information

2 associated with an LOS (Loss Of Signal) state based on a standardized common format, the DIA

3 hierarchy 22 converts the request into a device-A local format and then requests a device-A driver

4 24 to provide the LOS state information. Accordingly, the device-A driver 24 provides the LOS

5 state information based on the device-A local format to the DIA hierarchy 22. The DIA hierarchy

6 22 converts the LOS state information based on the device-A local format into the standardized

7 common format and then provides the LOS state information based on the standardized common

8 format to the high-order application 20. On the other hand, if a device A is changed to a device B,

9 a device-A driver is also changed to a device-B driver. The higher-order application 20 requests the

10 DIA hierarchy 22 to provide the LOS state information based on the standardized common format

11 irrespective of the changed device. Further, the DIA hierarchy 22 converts the request into a

12 device-B local format and then requests the device-B driver 26 to provide the LOS state information.

13 Accordingly, the device-B driver 26 provides the LOS state information based on the device-B local

14 format to the DIA hierarchy 22. The DIA hierarchy 22 converts the LOS state information based on

15 the device-B local format into the LOS state information based on the standardized common format

16 and then provides the LOS state information based on the standardized common format to the

17 higher-order application 20. As described above, the DIA hierarchy 22 arranged between the

18 lower-order device drivers 24 and 26 provides a mutual interface based on the standardized rule.

19 According to the change of the applications or devices, the verifications of the higher-order

20 applications and the device drivers are not needed.

1   **[0033]**   In order to provide the mutual interface based on the standardized rule between the

2   higher-order application 20 and the lower-order device drivers 24 and 26, the DIA hierarchy 22 in

3   accordance with the present invention reads materials from a DDCB (Device Driver Control Block)

4   and accesses the lower-order device driver using functions defined by the standardized rule. The

5   DDCB is defined to commonize the respective device drivers and then provides information

6   associated with the existence of a corresponding function and the position of a corresponding

7   function.

8   **[0034]**   In the embodiment of the present invention, only functions available in a corresponding

9   device driver among functions of a function block defined and standardized in international

10   organizations such as ITU/RFC (International Telecommunications Union/Request For Comments),

11   *etc.* is defined in a function table. The present invention employs all function blocks defined in

12   standardized documents made by international organizations for standardization such as ITU

13   (International Telecommunications Union), IETE (Internet Engineering Task Force), ETSI

14   (European Telecommunications Standardization Institute), ATM (Asynchronous Transfer Mode)

15   forum, ADSL (Asymmetrical Digital Subscriber Line) forum, *etc.* In the embodiment of the present

16   invention, only functions available in the corresponding device driver among functions of all

17   function blocks defined by the international organizations for standardization are re-defined in the

18   function table.

19   **[0035]**   In the embodiment of the present invention, the DIA hierarchy 22 uses device handler IDs

1 (identifiers) based on a standardized data format so that developers of higher-order applications

2 easily control the devices. The device handler ID has the standardized data format according to the

3 embodiment of the present invention and are unique identifiers corresponding to the respective

4 devices. The DIA hierarchy 22 provides the device handler IDs to the higher-order application 20

5 during the initialization of the corresponding device. The higher-order application 20 stores the

6 device handler IDs and calls a corresponding device using a corresponding device handler ID where

7 it is necessary to call the corresponding device. Accordingly, the DIA hierarchy 22 decides on the

8 basis of the device handler ID whether some device driver should be called, and then calls a device

9 driver according to the decision.

10 [0036] Hereinafter, a detailed description will be given of the device handler IDs being the unique

11 identifiers and a command control table generated and provided to the higher-order application 20

12 by the DIA hierarchy 22 with reference to Figs. 5 to 8. Further, a corresponding event table and a

13 profile of a corresponding device module mentioned in the embodiment of the present invention will

14 be described in detail with reference to Figs. 5 to 8.

15 [0037] Fig. 5 is a view illustrating a connection structure of a DCB (Device Control Block) after

16 an application user calls an API "Dia_InitDevice" of the DIA hierarchy 22 for the sake of the device

17 initialization. Fig. 6 is a view illustrating a final DCB in a level 1 initialization stage. Fig. 7 is a

18 view illustrating a connection structure of a DCB when an HDLC (High-level Data Link Control)

19 device having four ports corresponding to a level 2 is initialized. Fig. 8 is a view illustrating a

1 connection structure of a DCB when a channel is initialized.

2 **[0038]** First, the device handler IDs will be described in detail.

3 **[0039]** A device handler ID of a corresponding device generated by the DIA hierarchy 22 is

4 represented, for example, as a "DCB handlerId[1.0.0]" shown in Fig. 5. The device handler ID is

5 made up of three unsigned integers associated with a level 1, a level 2 and a channel, and has a.

6 unique value within the system. The device handler ID is made up of x1.x2.x3 where x1, x2 or x3

7 is an unsigned integer. The device handler ID is represented as "DCB handlerId[x1.x2.x3]". Here,

8 x1 is a value of the level 1 meaning a device ID, and x2 is a value of the level 2 meaning a logical

9 or physical group number of a corresponding device, and x3 is a value of the channel meaning a

10 channel number of a corresponding device or group. If values of x1, x2 and x3 are "0", it means that

11 there is no corresponding level or channel. When the device is initialized, an integer value of x1

12 sequentially increases from "1" as shown in Figs. 6 to 8. There cannot exist a handler having a value

13 of the level 1, *i.e.*, x1 = "0". This means that the value of the level 1 is not initialized.

14 **[0040]** As shown in Fig. 5, if the value of the level 1 is initialized, a DCB 32 containing elements

15 called by a device is dynamically assigned on the basis of the standardized rule in accordance with

16 the embodiment of the present invention. The elements include various pointers and function

17 pointers for performing the standardized rule in the DIA hierarchy 22. When the value of the level

18 1 is initialized, the elements of the DCB 32 include "*pHandler", "*fpInitDevice",

1   "*fpOpenChannel", "*fpCloseChannel", "*fpRead", "*fpWrite", "*fpReset", "*pControlTable",

2   "*pDDCB", "*pEventTable" and "*pAnchor". The "*pHandler" is a pointer for pointing a position

3   of a given initialization profile when a device is initialized, and the "*fpInitDevice" is a function

4   pointer used when a device is initialized. The "*fpOpenChannel" is a function pointer used when

5   a channel is open. The "*fpCloseChannel" is a function pointer used when a channel is closed. The

6   "*fpRead" is a function pointer used when data of an open channel is read. The "*fpWrite" is a

7   function pointer used when data of the open channel is written. The "*fpReset" is a function pointer

8   used when a device is reset. The "*pControlTable" is a pointer for pointing a position of a command

9   control table. The "*pDDCB" is a pointer for pointing a position of a device driver control table 36.

10   The "*pEventTable" is a pointer for pointing a position of an event table 38. The "*pAnchor" is a

11   pointer for pointing a next level.

12   [0041]   After all devices are initialized, the DIA hierarchy 22 provides generated device handler

13   IDs to the higher-order application 20. The higher-order application 20 initializes a corresponding

14   device and then stores only a given device handler ID. If a device handler is called, the DIA

15   hierarchy 22 receives the device handler ID, decides whether some device driver must be called, and

16   calls a device driver according to the decision.

17   [0042]   Next, a standardized command control table being a function table will be described.

18   [0043]   The standardized command control table employs function blocks and elements associated

1   with a corresponding function block defined by international organizations for standardization such

2   as ITU, IETE, ETSI, ATM forum, ADSL forum, *etc.* If there is an additional requirement, it can be

3   optionally added to the standardized command control table. For example, the present inventor

4   organized information relating to super high-speed communications as tables. The made tables are

5   stored in a corresponding memory portion. Further, if necessary, another function can be optionally

6   added.

7   **[0044]** As shown in Fig. 5, the standardized command control table 34 is positioned in a memory

8   portion pointed by the pointer of "*pControlTable" of a corresponding DCB. A command ID of the

9   command control table 34 is standardized and defined as a unique value. The command ID is

10   mapped to the function pointer of "*fpCommandFn" corresponding to a command function.

11   **[0045]** A structure of the command control table 34 according to the embodiment of the present

12   invention is as follows.

13   **[0046]** << Structure of standardized command control table >>

14   typedef enum

15   {

16   D_SDH_COMMAND_ID_START = 0x2000000

17   D_SDH_SPI_COMMAND_ID_START = 0x2000100

1   D_SDH_SPI_SET_ALS,

2   D_SDH_SPI_GET_ALS,

3   ....

4   D_SDH_COMMAND_ID_END

5   } DIA_COMMAND_ID_SDH_E;

6   **[0047]**   In the above-described structure of the standardized command control table 34, a

7   standardized command "typedef enum" such as "DIA_ COMMAND_ID_SDH_E" exists as

8   "DIA_COMMAND_ID_PDH_E", "DIA_COMMAND_ID_ATM_E", *etc.* on a field basis. Various

9   command functions can be arbitrarily added to the standardized command control table 34. The user

10   of the higher-order application can arbitrarily call a corresponding device driver through the

11   standardized command control table 34. Although a device driver is changed, a corresponding

12   standardized command remains as it is. Only a function pointer is varied when the corresponding

13   device driver carries out a corresponding command. The function pointer is provided to the DIA

14   hierarchy 22 by the device driver when a device is initialized. The command is called through an

15   API "Dia_Control" by the user of the higher-order application. When the user of the higher-order

16   application calls the API "Dia_Control" through a specific control command, only a device driver

17   positioned below the DIA hierarchy 22 is varied.

18   **[0048]**   Next, the device driver control table 36 according to the embodiment of the present

19   invention will be described with reference to Fig. 5. In Fig. 5, the device driver control table 36 is

1   a table pointed by the pointer of "*pDDCB". The device driver control table 36 contains function

2   pointers "*fp" for pointing positions of "InitLevel1", "InitLevel2", "OpenChannel", "CloseChannel",

3   a control table, an event table", *etc.*, *e.g.*, "*fpInitLevel1", "*fpInitLevel2", "*fpOpenChannel",

4   "*fpCloseChannel", "*fpControlTable (not shown)", "*fpEventTable (not shown)", *etc.* Further,

5   the device driver control table 36 contains information (not shown) indicating the number of

6   initialization functions, the number of ports and the number of channels associated with a device.

7   The device driver control table 36 is defined to commonize controls of the device drivers and

8   corresponds to a DDCB (Device Driver Control Block) for providing information associated with

9   the existence of a corresponding function and the position of a corresponding function. The device

10   driver control table 36 contains "*fpInitLevel1", "*fpInitLevel2", "*fpOpenChannel",

11   "*fpCloseChannel", "*fpControlTable (not shown)", "*fpEventTable (not shown)", *etc.* When a

12   device is initialized, the device driver control table 36 being the DDCB is located by the pointer of

13   "*pDDCB". The materials of "*fpInitLevel1", "*fpInitLevel2", "*fpOpenChannel",

14   "*fpCloseChannel", "*fpControlTable (not shown)", "*fpEventTable (not shown)", *etc.* are filled

15   as information of the assigned DCB 32.

16   **[0049]**   Next, an event table associated with the embodiment of the present invention will be

17   described.

18   **[0050]**   An event table 38 shown in Fig. 5 is a structure located at a position pointed by the pointer

19   of "*pEventTable" of a corresponding DCB being generated only if necessary. When the event table

1   is not used, it is indicated as a null. The event table 38 includes an event ID of "EventId" and an

2   event list structure pointer of "*pEventList". Each event list is a linked list as shown in Fig. 9.

3   Accordingly, a plurality of places can be notified of one event and a call-back function of "Call-back

4   Fn" can be called. The event table 38 initially includes the event ID of "EventId" and the pointer

5   of "*pEventList" is initially pointed as a null. When the user of the higher-order application 20 uses

6   a corresponding event, the higher-order application 20 can be connected to a pointer of an event list

7   structure using an API "Dia_Register".

8   **[0051]**   A structure of the event table 38 according to the embodiment of the present invention is

9   as follows.

10   **[0052]**   << Structure of event table>>

11   typedef struct

12   {

13   EXECFUNC   pCallBackFunc;

14   EXACTLY_INT32_T *pNextCallback;

15   } DIA_EVENT_CONFIG_T;

16   typedef struct

17   {

```
1          EXACTLY_UNIT32_T    eventId;

2              DIA_EVENT_CONFIG_T   *pEventList;

3          } DIA_EVENT_TABLE_T;
```

4   **[0053]**   A profile of a device module associated with the embodiment of the present invention will

5   be described.

6   **[0054]**   An input argument needed when the function pointer of "*fpInitDevice" of a corresponding

7   DCB 32 shown in Fig. 5 is carried out has a form of a structure pointer. In a special case, a structure

8   of a corresponding device can be defined. However, in a conventional case, a common structure

9   including a base address at the time of initializing a value of the level 1, a group ID corresponding

10   to the level 2 and a channel ID at the time of initializing a channel is used.

11   **[0055]**   A common structure of a profile of a device module according to the embodiment of the

12   present invention is as follows.

13   **[0056]**   << Common structure of profile of device module>>

```
14       typedef struct

15           {

16               EXACTLY_UNIT32_T  baseAddress1;
```

Page 17 of 36

```
1        EXACTLY_UNIT32_T  numExtraBaseAddresses;

2        EXACTLY_UNIT32_T  *pExtraBaseAddresses;

3   } DIA_BASE_ADDR_T;


4   typedef struct

5      {

6        DIA_BASE_ADDR_T   baseAddress;

7        EXACTLY_UNIT_32_T  *pUserDefine;

8   } DIA_COMMON_LEVEL1_PROFILE_T;


9   typedef struct

10     {

11       EXACTLY_INT32_T   groupNo;

12       EXACTLY_UNIT32_T  *pUserDefine;

13  } DIA_COMMON_LEVEL2_PROFILE_T;


14  typedef struct

15     {

16       EXACTLY_INT32_T   channelNo;

17       EXACTLY_UNIT32_T  *pUserDefine;

18  } DIA_COMMON_CHANNEL_PROFILE_T;
```

1   **[0057]**   Hereinafter, a detailed description of an operation according to the embodiment of the

2   present invention is as follows.

3   **[0058]**   When the higher-order application 20 calls a function of a function block to be used, the

4   DIA hierarchy 22 identifies the existence of a corresponding function from a function table using the

5   DDCB having information indicating the existence and positions of corresponding function tables.

6   If the corresponding function tables exist, the corresponding function is called. To call the function,

7   the DIA hierarchy 22 informs the higher-order application 20 of a device handler ID when a

8   corresponding device is initialized and then the higher-order application 20 accesses a lower-order

9   device driver using the device handler ID.

10   **[0059]**   First, an operation at the time of initializing a device is as follows.

11   **[0060]**   When the user of the higher-order application 20 calls an API "Dia_InitDevice", the DIA

12   hierarchy 22 dynamically assigns the DCB 32 containing pointers associated with each of function

13   blocks (*e.g.*, a control table, an event table, *etc.*) used by the higher-order application 20 through the

14   DDCB having information indicating corresponding function tables.

15   **[0061]**   After the API "Dia_InitDevice" is called, a connection of the DCB will be described in

16   detail with reference to Figs. 5 to 8. According to a basic flow form, an addition of a database

17   structure will be described.

[0062]   (1) Level 1 initialization

[0063]   A device to be initialized has a level 1, a level 2 and a channel. When the level 1 is initialized, the DCB 32 being a highest-order block of the device is dynamically assigned. Thereafter, the level 2 of the device is pointed by an anchor pointer of "*pAnchor" of the DCB 32 associated with the level 1. Since the number of devices used in each card is a known value, it is assumed that a highest-order database 40 is defined with a pointer of "*DCB" indicating a device ID shown in Fig. 5 as a global variable and the DCB 32.

[0064]   A form when the level 1 is initialized is the same as Fig. 5. Further, in a level 1 initialization stage, a final DCB is shown in Fig. 6. Whenever the DCB 32 shown in Fig. 5 is generated, the DIA hierarchy 22 returns a unique device handler ID value of "DCB HandlerId[x1.x2.x3]" corresponding to each DCB 32 to the higher-order application 20 as shown in Fig. 6. That is, the DIA hierarchy 22 gives the device handler ID value of "DCB HandlerId[x1.x2.x3]" while sequentially increasing a value of x1 from "1" according to an initialization sequence of the level 1 of a device. Referring to Fig. 6, a device handler ID value of an HDLC (High level Data Link Control) returns to "DCB HandlerId[1.0.0]". A device handler ID value of a LAN (Local Area Network) returns to "DCB HandlerId[3.0.0]". A device handler ID value of a UTOPIA (Universal Test & Operations Physical layer Interface for ATM) returns to "DCB HandlerId[4.0.0]".

[0065] The higher-order application 20 should manage the returned device handler ID value of "DCB HandlerId[x1.x2.x3]". The user of the higher-order application 20 can call a corresponding function using the returned device handler ID value of "DCB HandlerId[x1.x2.x3]". Further, since the device handler ID corresponding to a device is given on the basis of the device initialization sequence, the user of the higher-order application 20 should identify any device corresponding to the returned device handler ID. For example, when there are three HDLC1, HDLC2 and HDLC3 devices, the x1 value of the HDLC3 device becomes "1" when the HDLC3 device is initialized.

[0066] (2) Level 2 initialization

[0067] The DIA hierarchy 22 can carry out the level 2 initialization after the level 1 initialization. At this time, used information includes information of the allowable number (the number of ports) in the level 2 associated with the DDCB (having a unique initialization function, and information relating to the number of ports or channels). For example, an anchor is assigned so that the level 2 initialization can be carried out by referring to the number of ports associated with a corresponding device. Further, the DIA hierarchy 22 designates an address of the DCB 32 generated at an anchor of the level 1 so that the DCB 32 necessary for the level 2 can be referred to.

[0068] Fig. 7 is a view illustrating a connection structure of a DCB when an HDLC device having four ports corresponding to a level 2 is initialized. Referring to Fig. 7, anchors of "Anchor1", "Anchor2", "Anchor3" and "Anchor4" assigned to the four ports corresponds to device handler ID

1 values of "DCB HandlerId[1.1.0]", "DCB HandlerId[1.2.0]", "DCB HandlerId[1.3.0]" and "DCB

2 HandlerId[1.4.0]", respectively. For example, if there are the four ports of "port1", "port2", "port3"

3 and "port4", the anchors of "Anchor1", "Anchor2", "Anchor3" and "Anchor4" are assigned to the

4 four ports of "port1", "port2", "port3" and "port4", respectively. The device handler ID values of

5 "DCB HandlerId[1.1.0]", "DCB HandlerId[1.2.0]", "DCB HandlerId[1.3.0]" and "DCB

6 HandlerId[1.4.0]" can be given to the four ports of "port1", "port2", "port3" and "port4",

7 respectively.

8 **[0069]** (3) Channel initialization

9 **[0070]** A DCB associated with a channel is generated only if necessary. Where the DCB is

10 associated with only a channel irrespective of the level 2, an anchor of "Anchor 0" is connected to

11 the DCB as shown in Fig. 8.

12 **[0071]** In a connection structure for channel initialization as shown in Fig. 8 similar to Fig. 7,

13 anchors identical with the number of channels are assigned and each anchor is connected to the

14 DCB. At this time, a device handler ID value is returned. A port corresponding to the anchor of

15 "Anchor 1" is associated with four channels of "channel 1", 'channel 2", "channel 3" and "channel

16 4". When the channel of "channel 3" is open, "DCB HandlerId[2.1.1]" for the open channel of

17 "channel 3" is returned according to an initialization sequence. The user of the higher-order

18 application 20 should manage a corresponding channel and a device handler ID value mapped to the

1   channel according to the device initialization sequence.

2   **[0072]** After the device initialization, the higher-order application 20 calls a function of a function

3   block using a device handler ID value of "DCB HandlerId[x1.x2.x3] being a standardized identifier.

4   Accordingly, the DIA hierarchy 22 identifies the existence of a corresponding function from a

5   function table of a corresponding device driver using a DDCB having information indicating

6   positions of corresponding function tables. If a corresponding function exists, the function is called.

7   **[0073]**   Fig. 10 is a view illustrating a case where a device driver is changed (or replaced) in

8   accordance with an embodiment of the present invention.

9   **[0074]** Referring to Fig. 10, although a device driver A is changed to a device driver B by the DIA

10   hierarchy 22, a device handler ID value of "DCB handlerId[1.0.0] for a device HDLC is never varied.

11   What is varied is that only a lower-order device driver and pointers (addresses) of the DCB 32 point

12   new devices under control of the DIA hierarchy 22. Accordingly, the higher-order application 20

13   can carry out the same functions although the lower-order device driver A is changed to the

14   lower-order device driver B.

15   **[0075]**   Figs. 11 to 14 are views illustrating a comparison of operations in a conventional art and

16   an embodiment of the present invention where a higher-order application or a lower-order device is

17   changed.

1   **[0076]**   Figs. 11 and 13 is views illustrating cases where a higher-order application is changed.

2   Fig. 11 is a view illustrating a case where a higher-order application (program) is changed in a

3   conventional art. Fig. 13 is a view illustrating a case where a higher-order application (program) is

4   changed in accordance with an embodiment of the present invention.

5   **[0077]**   First, referring to Fig. 11, when the higher-order application A is changed to the

6   higher-order application B in the conventional art, the higher-order application should be not only

7   changed but also the lower-order device driver should be changed. The reason why the higher-order

8   application and the lower-order device order should be changed is that structures and APIs associated

9   with the higher-order applications A and B are different and then the lower-order device driver

10 should provide changed forms of APIs that are requested. Accordingly, an operation of an addition,

11 deletion or correction should be applied to a pre-existing lower-order device driver. Changed

12 portions should be newly verified. Of course, the higher-order application B should be newly

13 verified.

14 **[0078]**   In order to make up for a disadvantage of the conventional art shown in Fig. 11, the DIA

15 hierarchy 22 is arranged between a higher-order application hierarchy and a lower-order device

16 driver hierarchy as shown in Fig. 13 in accordance with the embodiment of the present invention.

17 If the DIA hierarchy 22 is arranged between the higher-order application hierarchy and the

18 lower-order device driver hierarchy, the higher-order applications A and B can use the same control

19 command of Control A, Control B and Control C based on a DIA's standardized rule, although the

1   higher-order applications A and B are changed. The higher-order application does not directly

2   control the device driver, but indirectly controls the device driver through the DIA hierarchy 22 using

3   the control commands, which are included in devices. Referring to Fig. 13, the control commands

4   of Control A, Control B and Control C based on a standardized common format used by the

5   higher-order applications A and B are individually converted into other control commands of Control

6   A-1, Control B-1 and Control C-1 used in a device 1 by the DIA hierarchy 22. The control

7   commands of Control A-1, Control B-1 and Control C-1 are provided to a device-1 driver.

8   Therefore, the higher-order application is not especially dependant upon the lower-order device

9   driver, although it is changed.

10   [0079]   Figs. 12 and 14 are views illustrating a case where a higher-order application is not

11   changed, but a lower-order device is changed. Fig. 12 is a view illustrating a case where a

12   lower-order device is changed in a conventional art. Fig. 14 is a view illustrating a case where a

13   lower-order device is changed in accordance with an embodiment of the present invention.

14   [0080]   First, referring to Fig. 12, where a higher-order application is not changed, but a

15   lower-order device is changed in the conventional art, an API of the lower-order device driver is

16   changed. Similarly, to use the changed API of the lower-order device driver, the higher-order

17   application is changed. That is, a part of the higher-order application should be deleted or corrected

18   or another part should be added to it. If so, the higher-order application as well as the changed

19   lower-order device driver should be re-verified.

[0081] On the other hand, if the DIA hierarchy 22 is arranged between the higher-order application hierarchy and the lower-order device driver hierarchy as shown in Fig. 14 in accordance with the embodiment of the present invention, the higher-order application has not to be changed and only the lower-order device driver is changed on the basis of the DIA hierarchy 22. Accordingly, only the verification of the lower-order device driver is needed. Referring to Fig. 14, the control commands of Control A, Control B and Control C based on a standardized common format used by the higher-order application A are individually converted into other control commands of Control A-1, Control B-1 and Control C-1 included in a device 1 by the DIA hierarchy 22. The control commands of Control A-1, Control B-1 and Control C-1 are provided to a device-1 driver. However, if the lower-order device driver is changed from a device-1 driver to a device-2 driver, the device-2 driver informs the DIA hierarchy 22 that control commands used in a device 2 are Control A-2, Control B-2 and Control C-2. Thereafter, the DIA hierarchy 22 individually converts the control commands of Control A, Control B and Control C based on the standardized common format used by the higher-order application A into other control commands of Control A-2, Control B-2 and Control C-2 used in a device 2.

[0082] The present invention prevents a higher-order application hierarchy and a lower-order device driver hierarchy from directly accessing each other by arranging a DIA (Device Independent Access) hierarchy between the higher-order application hierarchy and the lower-order device driver hierarchy in a communication system. Accordingly, the higher-order application hierarchy and the lower-order device driver hierarchy can access the lower-order device driver hierarchy and the

1   higher-order application hierarchy via the DIA on the basis of the DIA's standardized rule,

2   respectively. Because the higher-order application hierarchy and the lower-order device driver

3   hierarchy accesses the lower-order device driver hierarchy and the higher-order application hierarchy

4   on the basis of the DIA's standardized rule, respectively, a period of time required for the

5   development of products, and costs of product development can be reduced, and efficiency of

6   product development can be improved.

7   [0083]   Although the preferred embodiments of the present invention have been disclosed for

8   illustrative purposes, those skilled in the art will appreciate that various modifications, additions and

9   substitutions are possible, without departing from the scope of the invention. Therefore, the present

10   invention is not limited to the above-described embodiments, but the present invention is defined

11   by the claims, which follow along with their full scope of equivalents.